

TABLE DES MATIÈRES

I. Présentation du contexte ICGO.....	2
II. Présentation du projet.....	2
III. Analyse.....	3
A. Modélisation des données.....	3
B. Les traitements.....	4
IV. Réalisation technique.....	6
A. Architecture de l'application.....	6
B. Description des classes métiers de BiblioICGO.....	6
1. Module.....	7
2. Session.....	8
3. Inscription.....	8
4. Stage.....	9
5. StageEtale.....	10
C. L'accès à la base de données.....	13
D. L'accès à l'application.....	13
E. Programmation des IHM simples (agences, stagiaires, compétences, modules)	14
F. Principe de fonctionnement des IHM composés (stages, formateurs).....	18
G. Les sessions de stage.....	20
H. Les inscriptions aux sessions de stage.....	21

I. Présentation du contexte ICGO

ICGO (Institut Claude Gaston Octave) est une organisation se situant à Dijon, qui a pour objectif d'assurer des stages de formation informatique, pour tout le monde. Dirigé par Claude, Gaston et Octave. Il ont pour objectif de se développer pour être plus proche de leur clientèle, et ont donc ouvert, une agence à Beaune et une à Auxerre. Le siège social se situe à Dijon.

II. Présentation du projet

➤ Étude de l'existant

Leur site se trouve à Dijon, là où se trouve leur réseau administratif, Les serveurs ont pour but d'assurer des fonctions basiques (DNS, annuaire et gestion centralisé des environnement) et la communication (Messagerie, agenda partagé...) en plus des fonctions générique de l'entreprise (Comptabilité, RH, facturation...).

Ce sont des données stratégiques qui ne doivent ni fuir, ni être détruit, des sauvegardes régulières sont effectuées, pour garantir une sécurité maximale. Un réseau pédagogique indépendant est installé afin de permettre le bon déroulement des informations.

➤ Critique de l'existant

L'application créée lors de la création de l'entreprise devient obsolète au fil du temps, et elle n'a pas toutes les fonctionnalités requises au bon fonctionnement de l'entreprise, en effet, il y a des enseignements communs à plusieurs stages, cela complique donc la mise à jour des supports.

➤ Expression des besoins

Il y a besoin d'une restructuration complète de la base de données. L'application doit pouvoir enregistrer des catalogues de formations, l'ensemble des formateurs ainsi que les inscriptions des stagiaires aux sessions de stages, gérer des agences. Application accessible uniquement aux membres de l'entreprise. Cette application doit cependant respecter certaines contraintes comme une architecture par couches, nommage de fichier particulier.

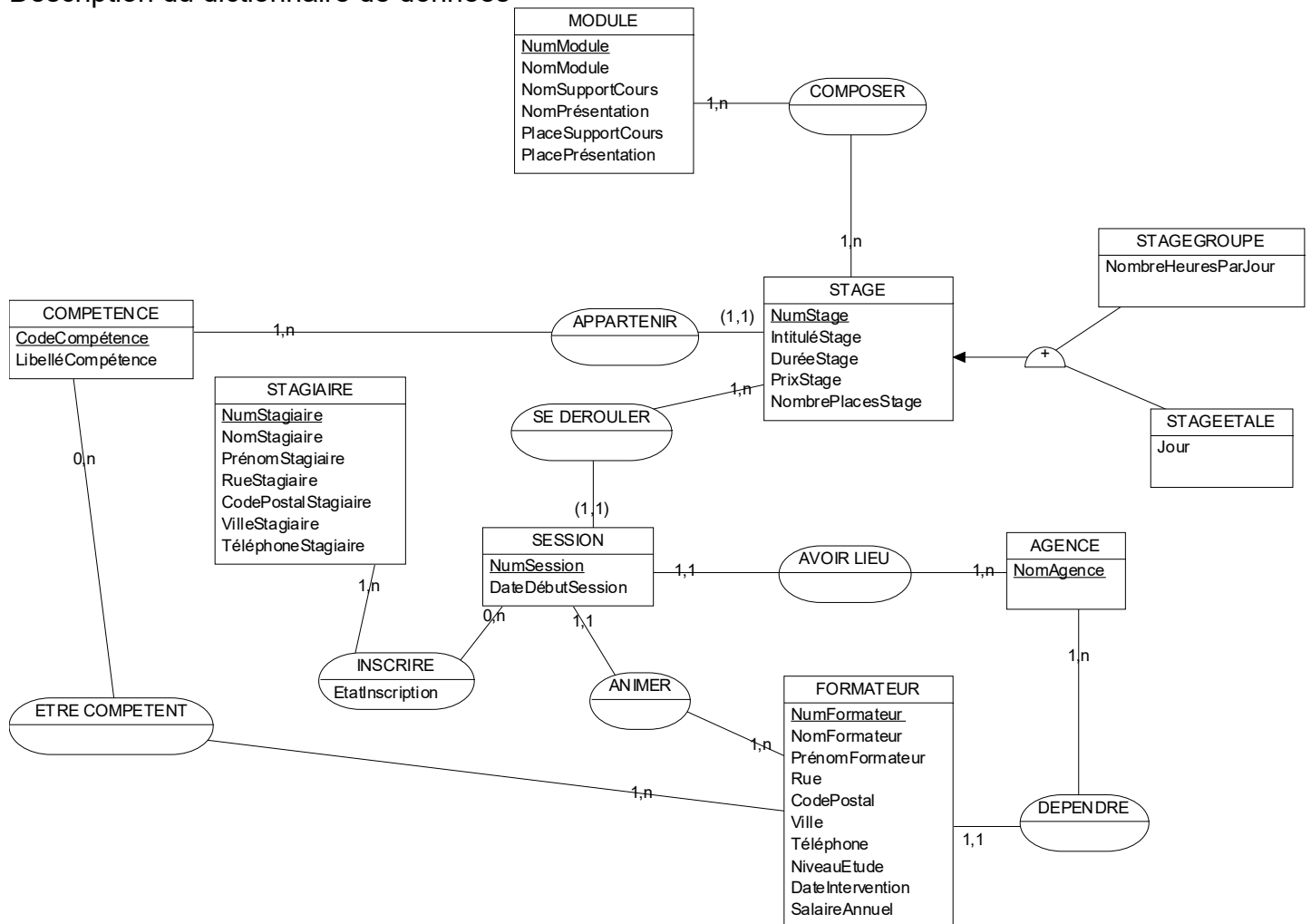
➤ Proposition de solution

Restructuration complète de la base de données, Application développée en C#, associée à la base de données SQL SERVER, mise en place d'une authentification, cette application respectera la charte graphique les différentes contraintes (nom du document, variables et paramètres...)

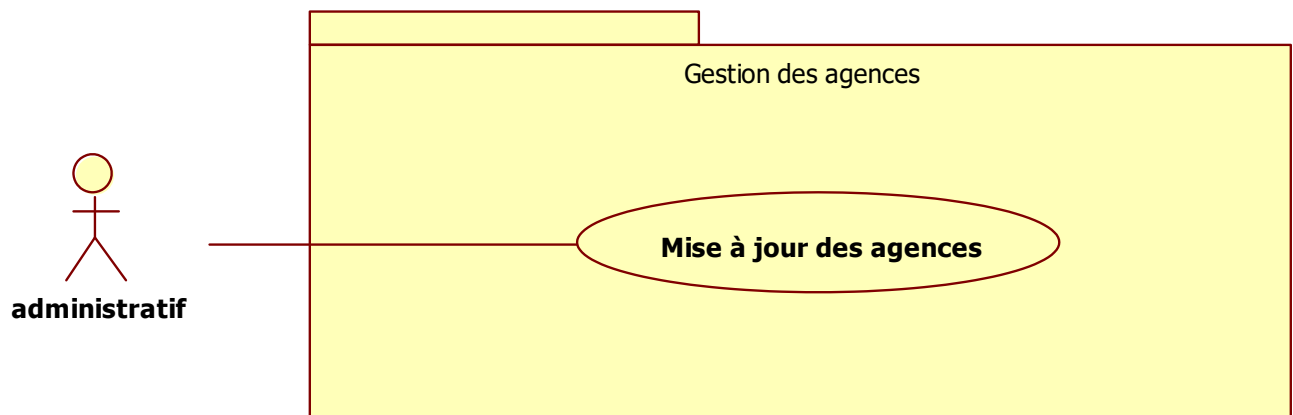
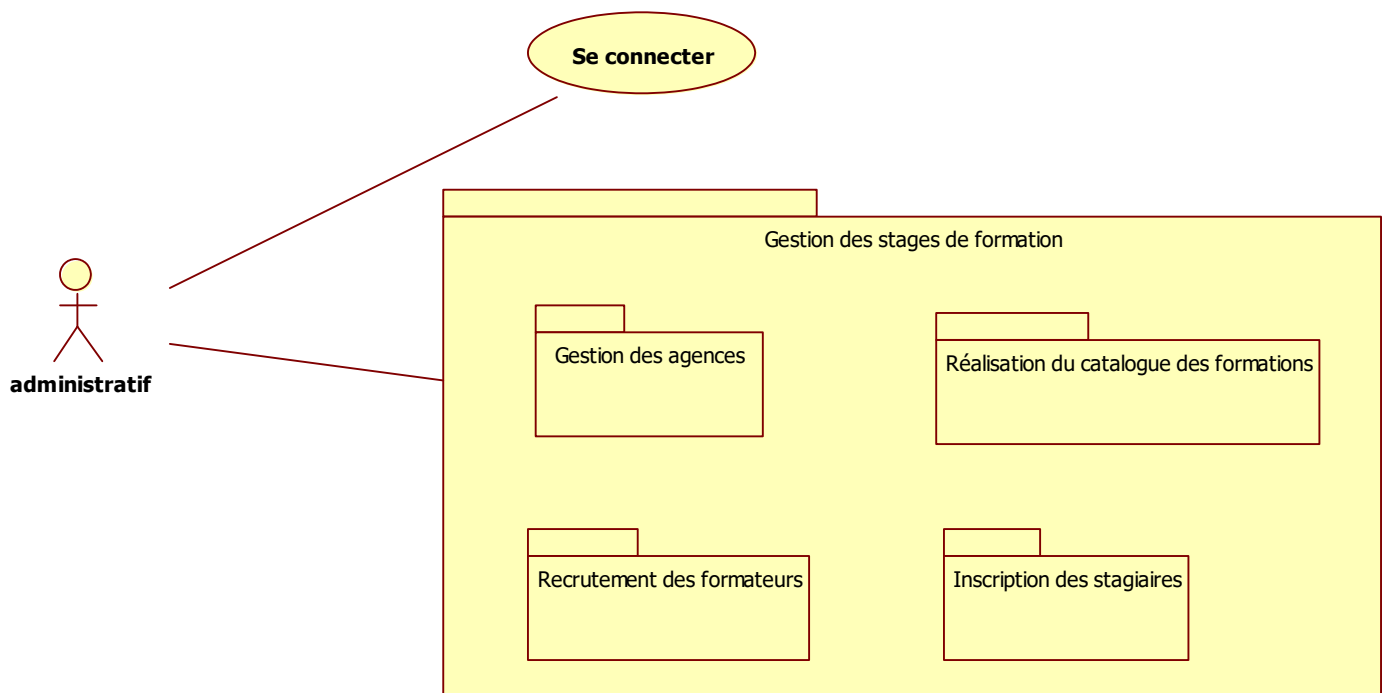
III. Analyse

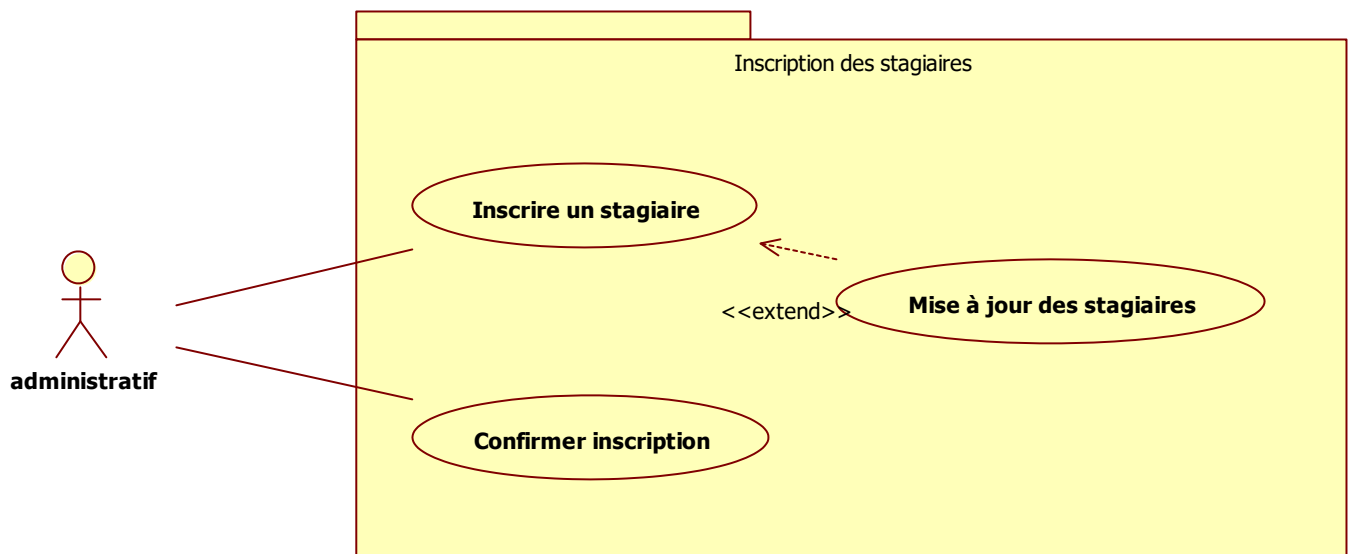
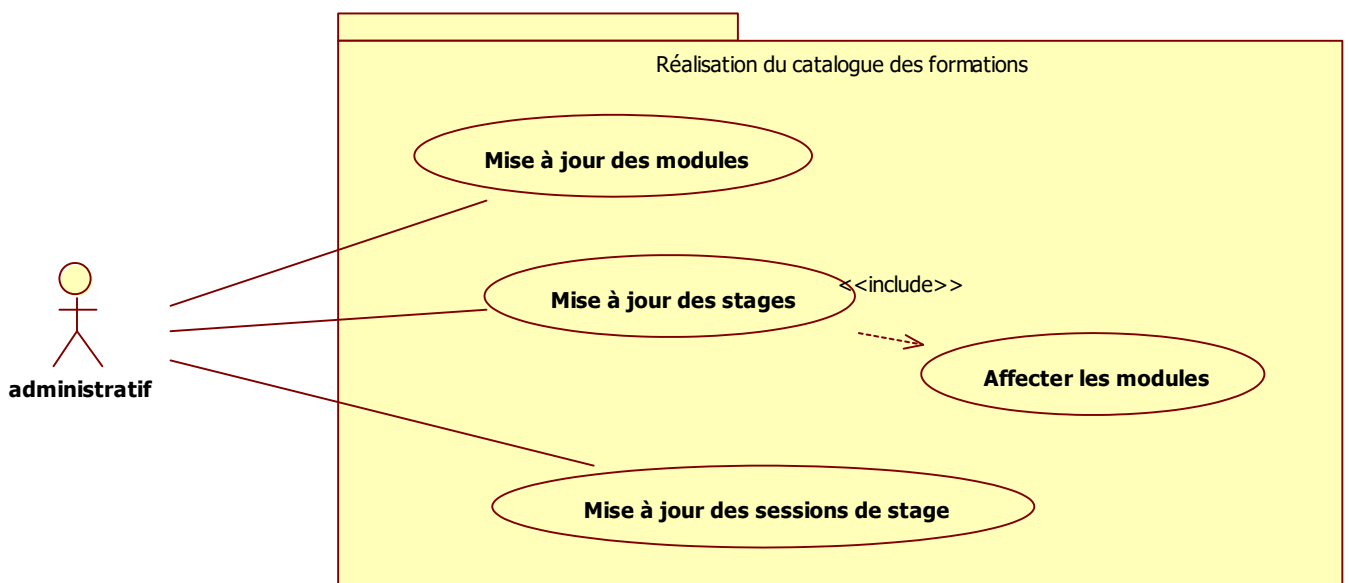
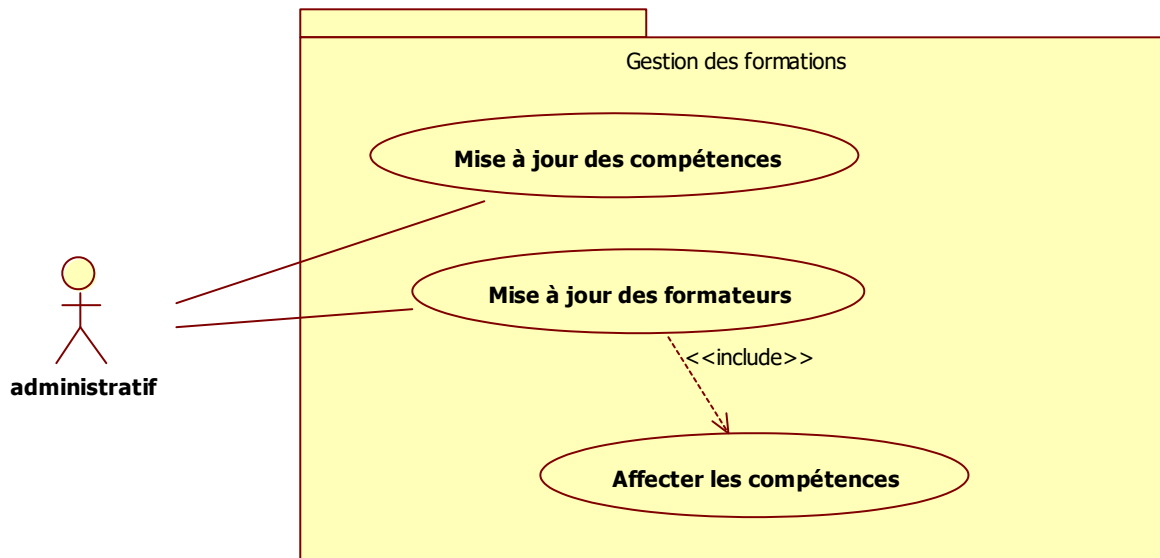
A. Modélisation des données

Description du dictionnaire de données



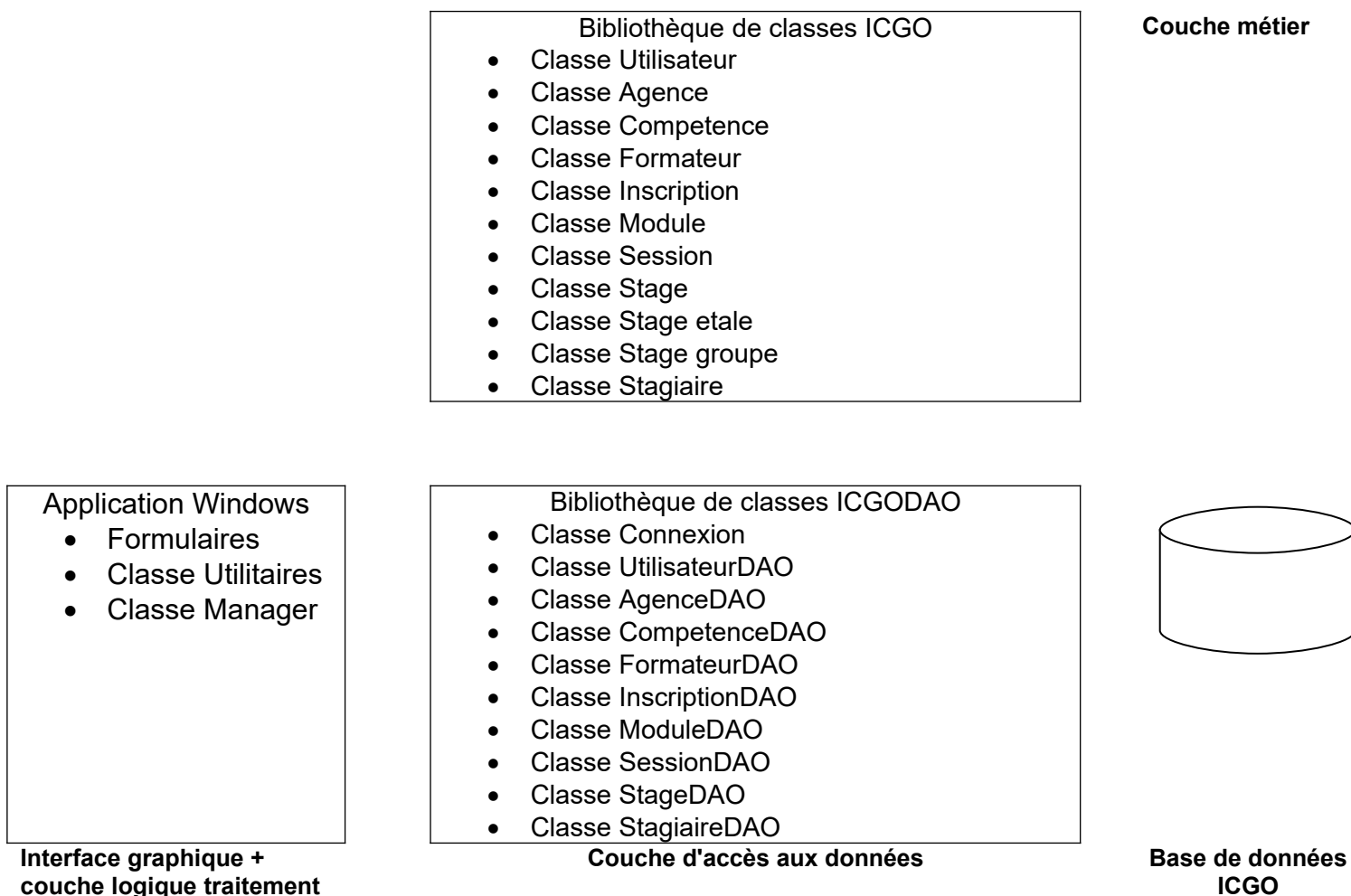
B. Les traitements





IV. Réalisation technique

A. Architecture de l'application



3 projets sont définis :

- **BibliolCGO** : bibliothèque de classes métiers.
- **BibliolCGODAO** : bibliothèque de classes permettant l'accès à la base de données, L'accès à la base de données s'effectue par l'intermédiaire d'une classe **Connexion**.
- **ProjetICGO** : application gérant les IHM et les classes Utilitaires et Manager.

B. Description des classes métiers de BibliolCGO

BibliolCGO est la 1ère bibliothèque de classe du projet. Elle sert à initialiser toutes les classes du projet, avec des constructeurs, accesseurs Get (récupérer une variable) et Set (attribuer une nouvelle valeur à la variable). Ici, nous avons 6 classes principales :

- Module
- Session
- Stage
- Stage étalé
- Stage groupé
- Inscription

Chaque classe a ses spécificités :

1. Module

Les variables privées de la classe Module sont toutes les caractéristiques propres au modules de stage, nous avons :

- numModule (entier) : le numéro de module.
- nomModule(chaîne) : le nom du module.
- nomSupportCours(chaîne) : le nom du support utilisé pendant le cours.
- nomPresentation(chaîne) : le nom de la présentation.
- placesupportCours(chaîne) : emplacement de stockage du support de cours.
- placePresentation(chaîne) : emplacement de stockage de la présentation.
- LesStages (liste de Stage) : Tous les stages contenant ce module.

+constructeurs module() : constructeur par défaut du module

+constructeurs module (entier unnumModule, chaîne unnomModule, chaîne unnomSupportCours, chaîne unnomPresentation, chaîne uneplaceSupportCours) : constructeur qui utilise les variables pour créer un module

+constructeurs module(entier unnumModule, chaîne unnomModule, chaîne unnomSupportCours, chaîne unnomPresentation, chaîne uneplaceSupportCours, liste de Stage Stage) : Ajoute la liste de stages au module.

GetnumModule() : retourne numModule

setnumModule(entier unnumModule) : modifie

getnomModule() : retourne nomModule

setnomModule(string unnomModule) : modifie nomModule

getnomSupportCours() : retourne nomSupportCours

setnomSupportCours(chaîne unnomSupportCour) : modifie nomSupportCours

getnomPresentation() : retourne le nom de la présentation

setnomPresentation(chaîne unnomPresentation) : modifie le nom de la présentation

getplaceSupportCours() : retourne l'emplacement de stockage du support de cours

setplaceSupportCours(chaîne uneplaceSupportCourts) : modifie l'emplacement de stockage du support de cours

getplacePresentation() : retourne l'emplacement de stockage de la présentation

setplacePresentation(chaîne uneplacePresentation) : modifie l'emplacement de stockage de la présentation

getlesStage() : retourne la liste de stages du module

setlesStage(liste de Stage) : modifie la liste de stages du module

2. Session

Les variables privées de la classe Session sont toutes les caractéristiques propres aux sessions :

- numSession(entier) : le numéro de session.
- dateSession(date) : la date de la session
- leStage(Stage) : le stage concerné par la session
- leFormateur(Formateur) : le formateur de la session
- lAgence(Agence) : l'agence de la session
- LesStagiaires(liste de stagiaire) : la liste des stagiaire de la session

- +constructeur Session() constructeur par défaut de la session
- +constructeur Session(entier numero, date date, Stage stage, Formateur formateur, Agence agence) : constructeur qui utilise les variables pour créer une session
- +constructeur Session(entier numero, date date, Stage stage, Formateur formateur, Agence agence, list de Stagiaires stagiaires) : Ajoute la liste de stagiaire au session.

- GetNumeroSession() : retourne le numéro de la session
- setNumeroSession(entier unNumero) : modifie le numéro de la session
- getDateSession() : retourne la date de la session
- setDateSession(date uneDate) : modifie la date de la session
- getLeStage() : retourne le stage concerné par la session
- setLeStage(Stage unStage) : modifie le stage concerné par la session
- getLeFormateur() : retourne le formateur de la session
- setLeFormateur(Formateur unFormateur) : modifie le formateur de la session
- getLAgence() : retourne l'agence de la session
- setLAgence(Agence uneAgence) : modifie l'agence de la session
- getLesStagiaires() : retourne les stagiaires inscrits a la session
- setLesStagiaires(liste de stagiaire desStagiaires) : modifie les stagiaires inscrits a la session

3. Inscription

Les variables privées de la classe Inscription sont toutes les caractéristiques propres aux inscriptions :

- laSession(Session) : la session rejoin lors l'inscription
- leStagiaire(Stagiaire) : le stagiaire qui a fait son inscription
- etatInscription(chaine) : l'etat de l'inscription

- +constructeur Inscription() : constructeur par défaut de l'inscription
- +constructeur Inscription(Session uneSession, Stagiaire unStagiere, chaîne unetatInscription) : constructeur qui utilise les variables pour créer une inscription.

- GetSession() : retourne la session concernée par l'inscription
- setSession(Session uneSession) : modifie la session concernée par l'inscription
- getStagiaire() : retourne le stagiaire inscrit
- setStagiaire(Stagiaire unStagiaire) : modifie le stagiaire inscrit
- getetatInscription() : retourne l'état de l'inscription
- setetatInscription(chaine unetatInscription) : modifie l'état de l'inscription

4. Stage

Les variables privées de la classe stage sont toutes les caractéristiques propres aux stage :

- numStage(entier) : le numero du stage
- nomStage(chaîne) : le nom du stage
- duree(entier) : la duree du stage
- prix(chaîne) : prix du stage
- nbPlaces(entier) : nombre de place du stage
- laCompetence(Competence) : compétence acquis durant le stage
- lesModules(liste des module) : les module présent durant le stage

+constructeur Stage() : constructeur par défaut

+constructeur Stage(entier unNumStage, chaîne unNomStage, entier uneDuree, chaîne unPrix, entier unNbPlace, competence uneCompetence) : constructeur qui utilise les variables pour créer un stage

+constructeur stage(entier unNumStage, chaîne unNomStage, entier uneDuree, chaîne unPrix, entier unNbPlace, Competence uneCompetence, liste de module desModules) : Ajoute la liste de module au stage.

GetNumStage : retourne le numero du stage

SetNumStage(entité value) : modifie le numero de stage

Get NomStage() : retourne le nom de stage

SetNomStage(chaîne value) : modifie le nom de stage

Get Duree() : retourne la durée du stage

SetDuree(entité value) : modifie la durée du stage

Get Prix() : retourne le prix du stage

SetPrix(chaîne value) : modifie le prix du stage

GetNbPlaces() : retourne le nombre de place du stage

SetNbPlaces(entité value) : modifie le nombre de place du stage

Get LaCompetence() : retourne la compétence du stage

SetLaCompetence(Competence value) : modifie la compétence du stage

GetLesModules() : retourne les module du stage

SetLesModules(liste de module value) : modifie les modules du stage

5. StageEtale

Les variables privées de la classe StageEtale hérité de Stage sont toutes les caractéristiques propres aux stage étalé :

-jour(chaîne) : la durée en jour du stage étalé

+constructeur StageEtale() : constructeur par défaut du stage étalé

+constructeur StageEtale(entier unNumStage, chaîne unNomStage, entier uneDuree, chaîne unPrix, entier unNbPlaces, Competence uneCompetence, chaîne unJour) : base(unNumStage, unNomStage, uneDuree, unPrix, unNbPlaces, uneCompetence) : constructeur qui utilise les variables pour créer un stage étalé

+constructeur StageEtale(entier unNumStage, string unNomStage, entier uneDuree, string unPrix, entier unNbPlaces, Competence uneCompetence, liste de Module desModules, string unJour) : base(unNumStage, unNomStage, uneDuree, unPrix, unNbPlaces, uneCompetence, desModules) : Ajoute la liste de module au stage étalé.

GetJour() : retourne la durée en jour du stage étalé

SetJour(chaîne value) : modifie la durée en jour du stage étalé

biblioStageGroupe

-nbHeures(chaîne) : le nombre d'heures du stage groupé

+constructeur StageGroupe() : constructeur par défaut de stage groupe

+public StageGroupe(entier unNumStage, chaîne unNomStage, entier uneDuree, chaîne unPrix, entier unNbPlaces, Competence uneCompetence, string unNbHeures) : constructeur qui utilise les variables pour créer un stage groupé

: base(unNumStage, unNomStage, uneDuree, unPrix, unNbPlaces, uneCompetence)

+public StageGroupe(entier unNumStage, chaîne unNomStage, entier uneDuree, chaîne unPrix, entier unNbPlaces, Competence uneCompetence, liste de Module desModules, chaîne unNbHeures) : base(unNumStage, unNomStage, uneDuree, unPrix, unNbPlaces, uneCompetence, desModules) :

Ajoute la liste de module au stage groupé.

GetnbHeures() : retourne le nombre d'heures du stage groupé

SetnbHeures(chaîne value) : modifie le nombre d'heures du stage groupé

Description des classes de BibliolCGODAO

Les classes DAO contiennent des méthodes liées aux classes normales, mais qui vont influencer sur la base de données. Elles utilisent les méthodes des classes "normales" pour récupérer les données qui seront envoyées dans la base de données. On a donc 4 classes DAO principales :

- **ModuleDAO** : La classe ModuleDAO va servir à gérer ce qui concerne les modules de stage dans la base de données. Elle est constituée de plusieurs méthodes :
 - **AjouterUnModule** : Ajoute un module dans la table MODULE de la base de données.
 - **ChargerLesModules** : Crée et retourne une liste contenant tous les modules existant dans la base de données.
 - **GetModule** : Retourne un module identifié par son ID.
 - **ModifierUnModule** : Met à jour un module (identifié par son ID) avec de nouvelles valeurs.
 - **SupprimerUnModule** : Supprime un module identifié par son ID.
 - **ChargerLesModulesDuStage** : Charge tous les modules liés à un stage (identifiées par l'ID du stage et l'ID de la compétence)
- **StageDAO** : La classe StageDAO va servir à gérer ce qui concerne les stages dans la base de données. Elle est constituée de plusieurs méthodes :
 - **AjouterUnStage** : Ajoute un stage dans la table STAGE de la base de données.
 - **ChargerLesStages** : Crée et retourne une liste contenant tous les stages existant dans la base de données.
 - **GetStage** : Retourne un stage identifié par son ID et celle de la compétence donnée par le stage.
 - **ModifierUnStage** : Met à jour un stage (identifié par son ID) avec de nouvelles valeurs.
 - **SupprimerUnStage** : Supprime un stage identifié par son ID.
 - **ChargerStagesDuModule** : Charge tous les stages liés à un module (identifiées par l'ID du module).
- **SessionDAO** : La classe SessionDAO va servir à gérer ce qui concerne les sessions dans la base de données. Elle est constituée de plusieurs méthodes :
 - **AjouterUneSession** : Ajoute une session dans la table SESSION de la base de données.
 - **ChargerLesSessions** : Crée et retourne une liste contenant toutes les sessions existant dans la base de données.
 - **GetSession** : Retourne une session identifiée par son ID.
 - **ModifierUneSession** : Met à jour une session (identifiée par son ID) avec de nouvelles valeurs.
 - **SupprimerUneSession** : Supprime une session identifiée par son ID.
 - **ChargerLesSessionsDuStagiaire** : Charge toutes les sessions de stages auquel le stagiaire est inscrit.
 - **ChargerLesSessionsNonChoisiesDuStagiaire** : Charge toutes les sessions de stages auquel le stagiaire n'est pas inscrit.
- **InscriptionDAO** : La classe InscriptionDAO va servir à gérer ce qui concerne les inscriptions aux sessions dans la base de données. Elle est constituée de plusieurs méthodes :
 - **AjouterUneInscription** : Ajoute une inscription dans la table INSCRIRE.
 - **ConfirmerUneInscription** : Attribue l'état définitif à une inscription (elle ne peut alors pas être modifiée).
 - **SupprimerUneInscription** : Supprime une inscription identifiée par l'ID de la compétence, du stage, de la session et du stagiaire.
 - **VerifierPlacesDisponibles** : Vérifie s'il reste assez de places pour inscrire un stagiaire.

Procédure stockée

Le but de cette procédure est de savoir s'il y a encore des places disponibles dans un stage, elle est utilisée dans la méthode VerifierPlacesDisponibles de InscriptionDAO.

Il y a 3 variables d'entrée :

@codeCompetence (type : char(3))

@noStage (type : int)

@noSession (type : int)

Et une variable de sortie :

@dispo (type : int)

C. L'accès à la base de données

Pour accéder à la base de données, l'application possède une classe connexion (fichier Connexion.cs). Cette dernière contient plusieurs méthodes permettant un accès à la base de données de A à Z :

- `getConnexion` : Retourne la connexion à la base de données utilisée.
- `OuvrirConnexion` : Ouvre une connexion à la base de données. On y spécifie le serveur et la base de données qu'on utilise.
- `FermerConnexion` : Ferme la connexion à la base de données après un traitement.
- `ExecuterRequete` : Exécute une requête dans la base de données, puis remplit une table avec les données renvoyées par la requête.
- `ExecuterRequeteMaj` : Utilisée pour les requêtes modifiant la base de données (suppression, mise à jour, insertion).

D. L'accès à l'application

Le serveur de gestion utilisé est SRV-TFS (Team Foundation Server) un système similaire à Github mais en local, installable sur un serveur. On utilise SSMS (Microsoft SQL Server Management Studio) pour accéder à la base de données.

La gestion des utilisateurs se fait à partir d'une classe Utilisateur et UtilisateurDAO, la classe DAO gère les requêtes avec la bases de données.

- `Utilisateur` : La classe Utilisateur est constituée de plusieurs méthodes :
 - `Get/Set NumUtilisateur` : Change la variable NumUtilisateur.
 - `Get/Set NomUtilisateur` : Change la variable NomUtilisateur.
 - `Get/Set PrenomUtilisateur` : Change la variable PrenomUtilisateur.
 - `Get/Set Login` : Change la variable Login.
 - `Get/Set MotPasse` : Change la variable MotPasse.
- `UtilisateurDAO` : La classe UtilisateurDAO est constituée de une méthode :
 - `VerifierUtilisateur` : Vérifie si un utilisateur existe et si sont mot de passe et le bon.

frmMenu.cs

└ Agence

└frmAgence.cs

└ Recrutement

└frmCompetence.cs

└frmFormateur.cs

└ Catalogue

└frmModule.cs

└frmStage.cs

└frmSessionStage.cs

└ Inscription

└frmInscription

└frmConfirmerInscription..cs

└frmStagiaire.cs

E. Programmation des IHM simples (agences, stagiaires, compétences, modules)

Cette IHM permet de gérer les différentes agences

Agences :

Page composée de :

Liste composée des différentes agences

Zone de saisie comprenant le nom de l'agence

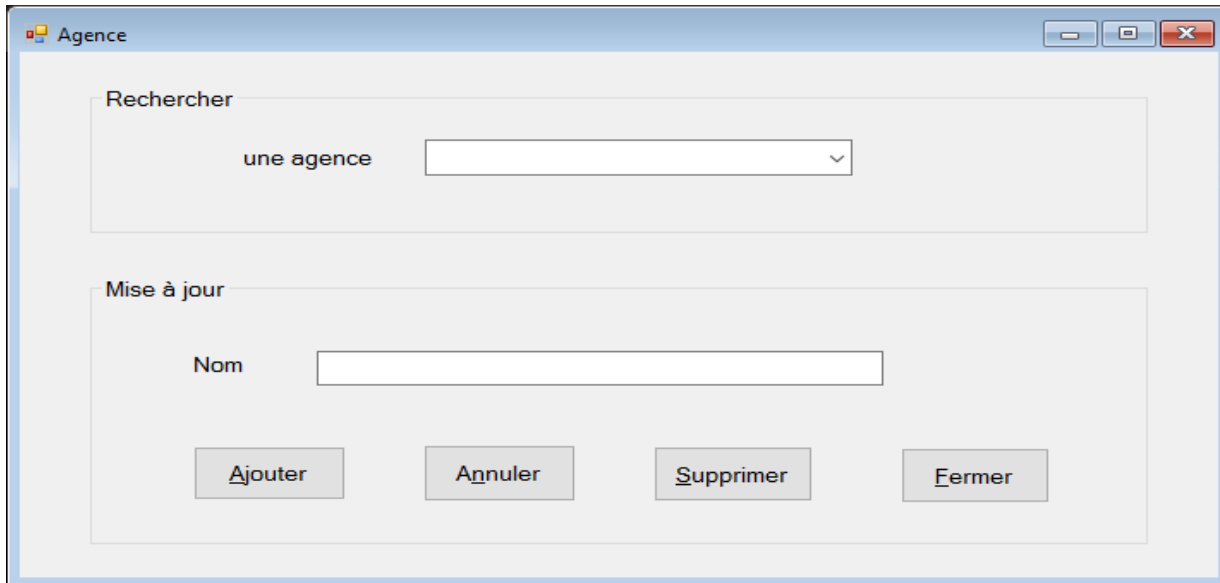
Boutons :

Ajouter : (`private void btnAjouter_Click`) Permet de faire l'ajout d'une agence

Annuler : (`private void btnAnnuler_Click`) Permet de vider les cases du formulaire

Supprimer : (`private void btnSupprimer_Click`) Permet de supprimer une agence sélectionnée

Fermer : (`private void btnFermer_Click`) Permet de fermer la fenêtre de formulaire



The screenshot shows a Windows application window titled "Agence". The window has a standard Windows title bar with minimize, maximize, and close buttons. The main content area is divided into two sections. The top section, labeled "Rechercher", contains a label "une agence" and a dropdown menu. The bottom section, labeled "Mise à jour", contains a label "Nom" and a text input field. At the bottom of the window, there are four buttons: "Ajouter", "Annuler", "Supprimer", and "Fermer".

Stagiaires :

Page composée de :

Liste composée des différent stagiaires

Des zone de texte pour:

- Un numéro de stagiaire
- Un nom de stagiaire
- Un prénom de stagiaire
- Une rue pour le stagiaire
- Un code postal pour le stagiaire
- Une ville pour le stagiaire
- Un numéro de téléphone pour le stagiaire

Boutons :

The screenshot shows a Java Swing window titled "Stagiaire" with standard Windows-style window controls (minimize, maximize, close). The window contains two main sections:

- Rechercher**: A section with a label "un stagiaire" and a text input field with a dropdown arrow.
- Mise à jour**: A section for updating a record, containing several text input fields:
 - Numéro stagiaire**: A single-line text field.
 - Nom**: A single-line text field.
 - Prénom**: A single-line text field.
 - Rue**: A single-line text field.
 - Code postal**: A single-line text field.
 - Ville**: A single-line text field.
 - Téléphone**: A single-line text field with a mask of four dashes (____).

Compétence :

Page composée de :

- D'une liste composée des différent compétences
- Une zone de texte pour un code compétences
- Une zone de texte pour un libellé de compétence
- Boutons :
 - Ajouter : (`private void btnAjouter_Click`) Permet de faire l'ajout d'une compétence
 - Annuler : (`private void btnAnnuler_Click`) Permet de vidé les case du formulaire
 - Modifier : (`private void btnModifier_Click`) Permet de modifier une compétence sélectionner
 - Supprimer : (`private void btnSupprimer_Click`) Permet de supprimer une compétence sélectionner
 - Fermer : (`private void btnFermer_Click`) Permet de fermer la fenêtre de formulaire

Compétence

Rechercher

une compétence

Mise à jour

Code compétence

Libellé

Ajouter Annuler Modifier Supprimer Fermer

Module :

Page composée de :

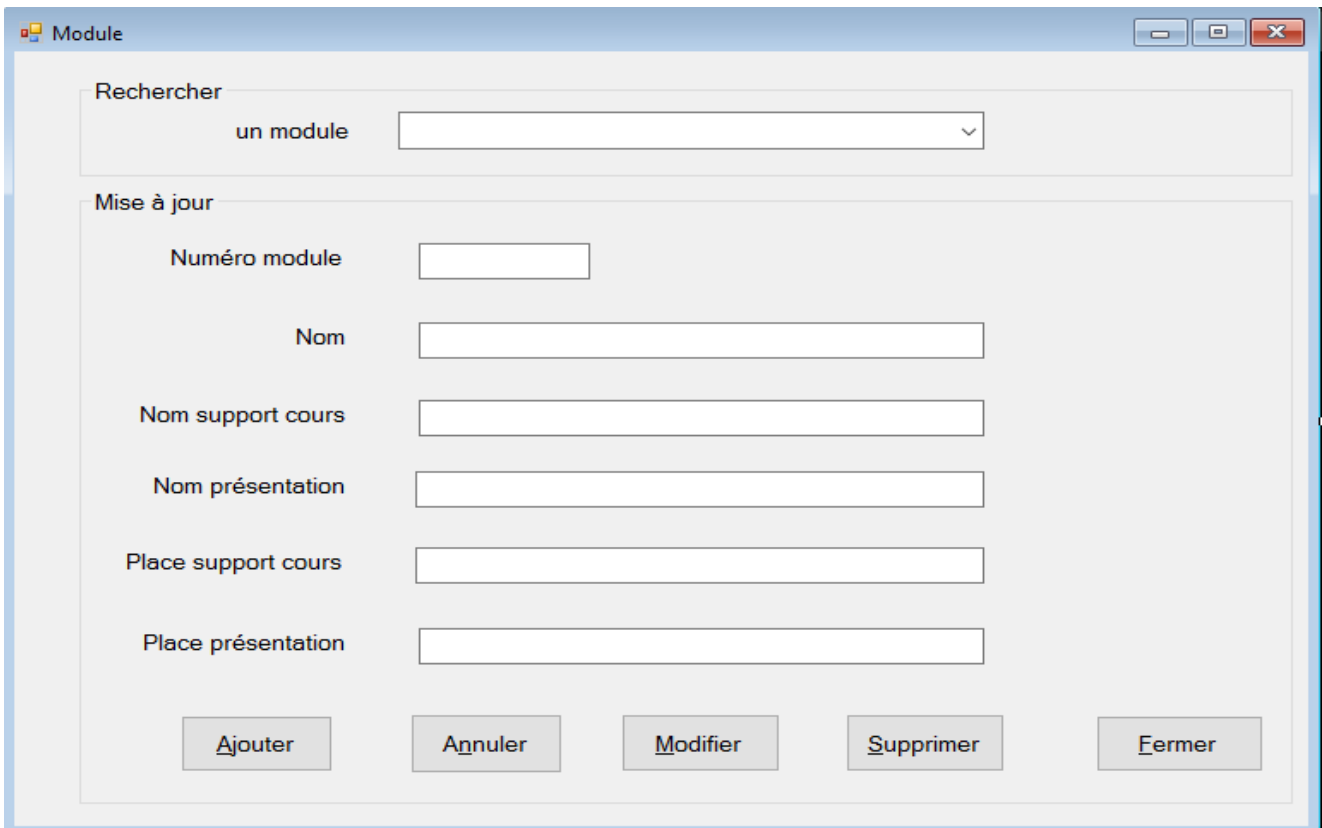
- D'une liste composée des différent module

Des zone de texte pour :

- Un numéro de module
- Un nom de module
- Un nom du support cours pour module
- Un nom de présentation pour module
- Un nom de place support cours pour module
- Un nom de place présentation pour module

Boutons :

- Ajouter : (`private void btnAjouter_Click`) Permet de faire l'ajout d'un module
- Annuler : (`private void btnAnnuler_Click`) Permet de vidé les case du formulaire
- Modifier : (`private void btnModifier_Click`) Permet de modifier un module sélectionner
- Supprimer : (`private void btnSupprimer_Click`) Permet de supprimer un module sélectionner
- Fermer :(`private void btnFermer_Click`) Permet de fermer la fenêtre de formulaire



Manager.cs a pour objectif de charge les différente liste de données.

Utilitaire.cs a pour objectif de séparer tout les donnés des chaînes et de le regrouper sous forme de tableaux.

F. Principe de fonctionnement des IHM composés (stages, formateurs)

Stages :

Page composée de :

- D'une liste composée des différents stage
- D'une liste composée des code compétence pour les stages

Des zone de texte pour :

- Un numéro de stage
- Un nom de stage
- Une durée (nombre de jours) des stage
 - Si option Etalé
 - Une liste comprenant les jour de la semaine
 - Si option Groupé
 - Une zone de texte pour le nombre d'heurs de stage
- Un Prix du stage
- Un nombre de places pour le stage

Boutons :

- Ajouter : (`private void btnAjouter_Click`) Permet de faire l'ajout d'un stage
- Annuler : (`private void btnAnnuler_Click`) Permet de vidé les case du formulaire
- Modifier : (`private void btnModifier_Click`) Permet de modifier un stage sélectionner
- Supprimer : (`private void btnSupprimer_Click`) Permet de supprimer un stage sélectionner
- Affecter les modules : (`private void btnAffecterModule_Click`) Permet de donner des modules a un stage
- Fermer :(`private void btnFermer_Click`) Permet de fermer la fenêtre de formulaire

The screenshot shows a Windows application window titled "Stage". The window contains a form with the following elements:

- Rechercher**: A label followed by a text box containing "un stage" and a dropdown arrow.
- Mise à jour**: A section header for the form fields.
- Code compétence**: A dropdown menu.
- Numéro stage**: A text box.
- Nom**: A text box.
- Durée (nombre de jours)**: A text box.
- Type stage**: A group box containing two radio buttons: **Etalé** and **Groupé**.
- Jour**: A dropdown menu, visible only when the "Etalé" radio button is selected.
- Nombre d'heures**: A text box, visible only when the "Groupé" radio button is selected.
- Prix**: A text box.
- Nombre de places**: A text box.
- Buttons**: A row of five buttons at the bottom: **Ajouter**, **Annuler**, **Modifier**, **Supprimer**, and **Affecter les modules**.

Formateur :

Page composée de :

- D'une liste composée des différents formateur
- D'une liste composée des différents agence pour les formateurs
- D'une liste composée des différents niveau d'étude pour les formateurs

Des zone de texte pour :

- Un numéro de formateur
- Un nom de formateur
- Un prénom de formateur
- Une rue pour le formateur
- Un code postal pour le formateur
- Une ville pour le formateur
- Un numéro de téléphone pour le formateur
- Une Date intervention pour le formateur
- Un salaire annuel pour un formateur

Boutons :

- Ajouter : (`private void btnAjouter_Click`) Permet de faire l'ajout d'un formateur
- Annuler : (`private void btnAnnuler_Click`) Permet de vidé les case du formulaire
- Modifier : (`private void btnModifier_Click`) Permet de modifier un formateur sélectionner
- Supprimer : (`private void btnSupprimer_Click`) Permet de supprimer un formateur sélectionner
- Affecter les modules : (`private void btnAffecterModule_Click`) Permet de donner des compétence a un formateur
- Fermer : (`private void btnFermer_Click`) Permet de fermer la fenêtre de formulaire

The screenshot shows a Windows application window titled 'Formateur'. It contains a search bar at the top with the label 'Rechercher' and a dropdown menu with the text 'un formateur'. Below this is a section titled 'Mise à jour' which includes a dropdown menu for 'Choisir une agence'. The main form area contains several input fields: 'Numéro formateur', 'Nom', 'Prénom', 'Rue', 'Code postal', 'Ville', 'Téléphone', 'Niveau d'étude' (with a dropdown), 'Date intervention' (with a date picker showing 'vendredi 8 avril 2022'), and 'Salaire annuel'. At the bottom of the window, there are six buttons: 'Ajouter', 'Annuler', 'Modifier', 'Supprimer', 'Affecter les compétences', and 'Fermer'.

G. Les sessions de stage

Sessions stage :

Page composée de :

- D'une liste composée des différents session de stage
- D'une liste composé des différents stage
- D'une liste composé des différents agence
- D'une liste composé des différent formateur
- D'une zone de texte pour un numéro de session de stage
- D'une zone de texte pour date session des session de stage

Boutons :

- Ajouter : (`private void btnAjouter_Click`) Permet de faire l'ajout d'une session stage
- Annuler : (`private void btnAnnuler_Click`) Permet de vidé les case du formulaire
- Modifier : (`private void btnModifier_Click`) Permet de modifier une session stage sélectionner
- Supprimer : (`private void btnSupprimer_Click`) Permet de supprimer une session stage sélectionner
- Fermer :(`private void btnFermer_Click`) Permet de fermer la fenêtre de formulaire

Session de stage

Rechercher

une session

Mise à jour

Choisir un stage

Numéro session

Date session

vendredi 8 avril 2022

Choisir une agence

Choisir un formateur

Ajouter Annuler Modifier Supprimer Fermer

H. Les inscriptions aux sessions de stage

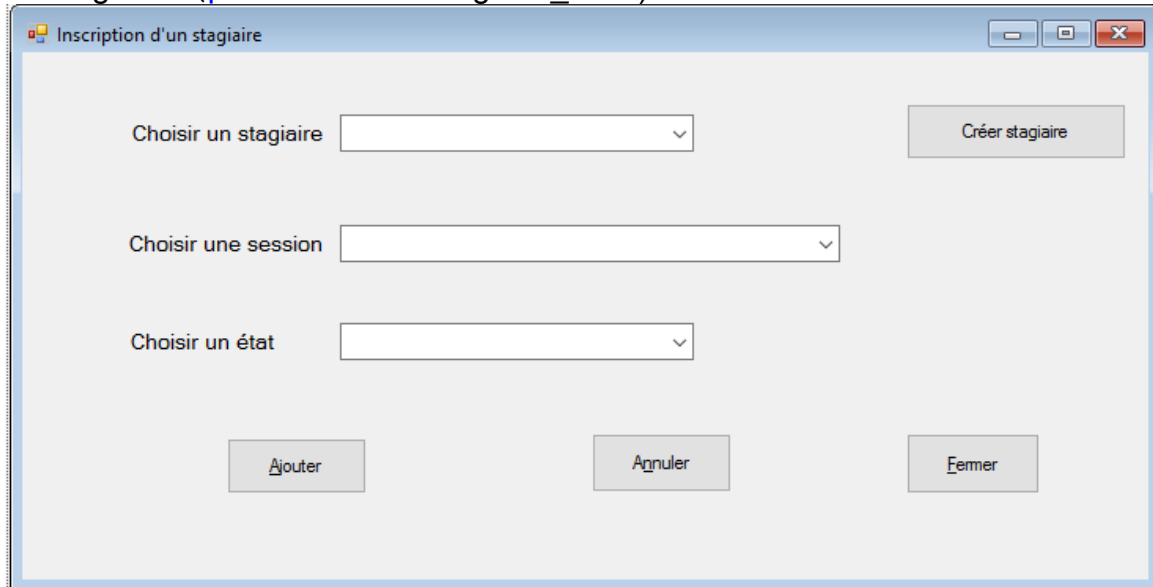
Inscription :

Page composée de :

- D'une liste composé de différent stagiaire
- D'une liste composé de différente sessions
- D'une liste composé de différent état

Boutons :

- Ajouter : (`private void btnAjouter_Click`) Permet de faire l'ajout d'une session stage
- Annuler : (`private void btnAnnuler_Click`) Permet de vidé les case du formulaire
- Fermer : (`private void btnFermer_Click`) Permet de fermer la fenêtre de formulaire
- Créer stagiaire: (`private void btnStagiaire_Click`) Permet d'affiche le formulaire d'un stagiaire



The screenshot shows a Windows application window titled "Inscription d'un stagiaire". The window has a standard Windows title bar with minimize, maximize, and close buttons. The main content area contains three dropdown menus, each with a label and a dropdown arrow: "Choisir un stagiaire", "Choisir une session", and "Choisir un état". To the right of the first dropdown is a button labeled "Créer stagiaire". At the bottom of the window are three buttons: "Ajouter", "Annuler", and "Fermer".

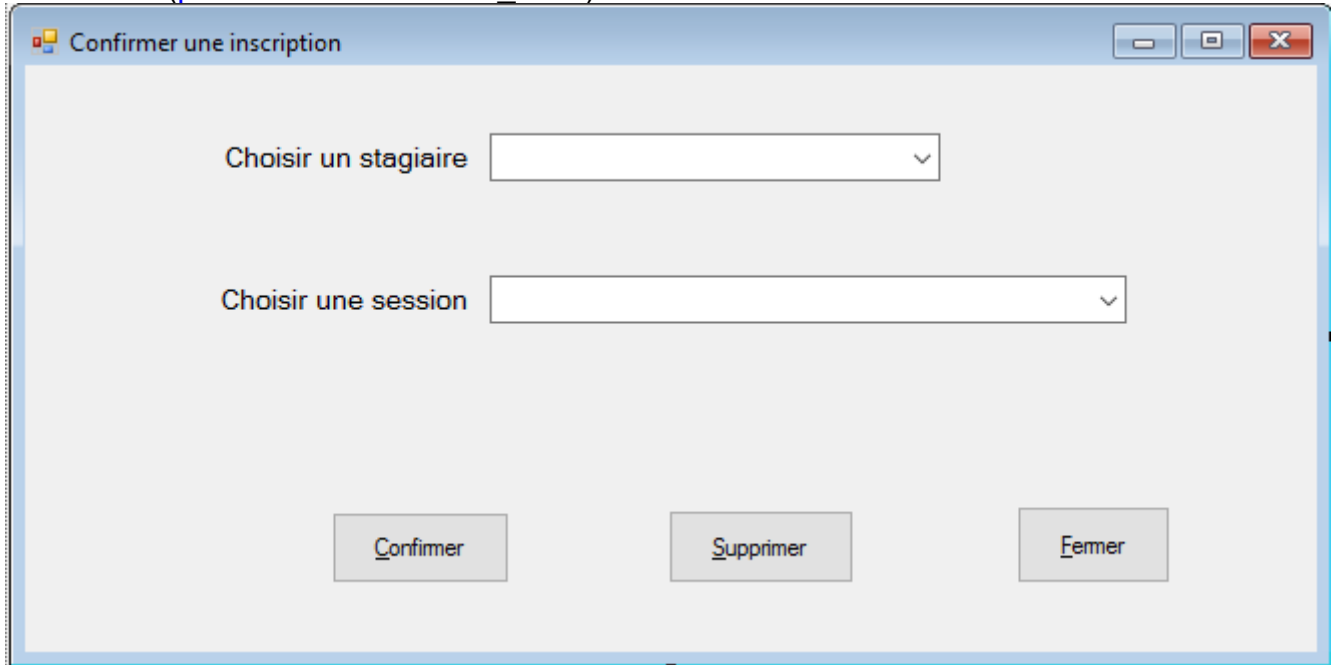
Confirmer Inscription :

Page composée de :

- D'une liste composé de différent stagiaire
- D'une liste composé de différente sessions

Boutons :

- Ajouter : (`private void btnAjouter_Click`) Permet de faire l'ajout d'une session stage
- Annuler : (`private void btnAnnuler_Click`) Permet de vidé les case du formulaire
- Fermer : (`private void btnFermer_Click`) Permet de fermer la fenêtre de formulaire



The screenshot shows a Windows application window with the title bar 'Confirmer une inscription'. Inside the window, there are two labels with corresponding dropdown menus: 'Choisir un stagiaire' and 'Choisir une session'. Below these, there are three buttons: 'Confirmer', 'Supprimer', and 'Fermer'. The window has standard Windows window controls (minimize, maximize, close) in the top right corner.